Package 'blends'

May 7, 2026

Title Blend Colours and Palettes

Version 0.1.1

Description Colour blend functions. These functions make it easier to blend colours and palettes using digital blend modes such as multiply, screen, and overlay.

License MIT + file LICENSE

URL https://github.com/davidhodge931/blends, https://davidhodge931.github.io/blends/

BugReports https://github.com/davidhodge931/blends/issues

Depends R (>= 4.1.0)

Imports rlang, scales

Suggests dplyr, ggplot2, jumble, spelling, testthat (>= 3.0.0), tidyr

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author David Hodge [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-3868-7501>)

Maintainer David Hodge <davidhodge931@gmail.com>

Repository CRAN

Date/Publication 2026-05-04 05:30:25 UTC

Contents

`colour_burn` *Blend colours and palettes using colour burn mode*

Description

Darkens the destination colour to reflect the source by increasing contrast. Produces deep, saturated results.

Usage

```
colour_burn(...)
```

Arguments

`...`	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
colour_burn("#FFA600FF", "#8991A1FF")
```

colour_dodge *Blend colours and palettes using colour dodge mode*

Description

Brightens the destination colour to reflect the source by decreasing contrast. Produces bright, washed-out results.

Usage

```
colour_dodge(...)
```

Arguments

...	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
colour_dodge("#FFA600FF", "#8991A1FF")
```

darken *Blend colours and palettes using darken mode*

Description

Darkens colours by selecting the darker of two colour values for each RGB channel. Useful for creating shadows or combining dark elements.

Usage

```
darken(...)
```

Arguments

...	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
darken("#FFA600FF", "#8991A1FF")
```

`difference` *Blend colours and palettes using difference mode*

Description

Subtracts the darker colour from the lighter. Identical colours produce black; white inverts the other colour.

Usage

```
difference(...)
```

Arguments

`...`	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
difference("#FFA600FF", "#8991A1FF")
```

exclusion *Blend colours and palettes using exclusion mode*

Description

Similar to difference but with lower contrast. Identical colours produce grey rather than black.

Usage

```
exclusion(...)
```

Arguments

`...`	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
exclusion("#FFA600FF", "#8991A1FF")
```

hard_light *Blend colours and palettes using hard light mode*

Description

Combines multiply and screen depending on the lightness of the first colour. Like overlay but the first colour controls whether darkening or lightening is applied.

Usage

```
hard_light(...)
```

Arguments

`...`	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
hard_light("#FFA600FF", "#8991A1FF")
```

`lighten` *Blend colours and palettes using lighten mode*

Description

Lightens colours by selecting the lighter of two colour values for each RGB channel. Useful for creating highlights or combining light elements.

Usage

```
lighten(...)
```

Arguments

`...`	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
lighten("#FFA600FF", "#8991A1FF")
```

multiply *Blend colours and palettes using multiply mode*

Description

Darkens colours by multiplying them together. Creates darker, more saturated results. Useful for creating shadows, darkening backgrounds, or adding depth.

Usage

```
multiply(...)
```

Arguments

`...`	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
multiply("#F0F0F0", "#808080")
multiply("#FF6B6B")
```

overlay *Blend colours and palettes using overlay mode*

Description

Combines multiply and screen depending on the lightness of the second colour. Values below 50% grey are multiplied (darkened); values above are screened (lightened).

Usage

```
overlay(...)
```

Arguments

`...`	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
overlay("#FFA600FF", "#8991A1FF")
```

`screen` *Blend colours and palettes using screen mode*

Description

Lightens colours by inverting, multiplying, then inverting again. Creates brighter results. Useful for creating highlights, lightening backgrounds, or adding luminosity.

Usage

```
screen(...)
```

Arguments

`...`	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
screen("#2C2C2C", "#808080")
screen("#4A4A4A")
```

`soft_light` *Blend colours and palettes using soft light mode*

Description

A softer version of hard light. Darkens or lightens depending on the first colour, but with a gentler effect than hard light.

Usage

```
soft_light(...)
```

Arguments

`...`	Either one or two colour/palette arguments: • If one argument: the colour or palette is blended with itself • If two arguments: the first is blended with the second Each argument can be a character vector of colours or a `scales::pal_*()` function

Value

Character vector of blended colours or a blending function.

Examples

```
soft_light("#FFA600FF", "#8991A1FF")
```

Index